DOCUMENT RESUME

ED 388 288 IR 017 437

AUTHOR	Recker, Margaret M.
TITLE	A Methodology for Analyzing Students' Interactions within Educational Hypertext.
PUB DATE	94
NOTE	7p.; In: Educational Multimedia and Hypermedia, 1994. Proceedings of ED-MEDIA 94--World Conference on Educational Multimedia and Hypermedia (Vancouver, British Columbia, Canada, June 25-30, 1994); see IR 017 359.
PUB TYPE	Reports - Research/Technical (143) -- Speeches/Conference Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Cognitive Processes; Computer Interfaces; *Computer System Design; Educational Technology; Feedback; *Hypermedia; *Interaction; Learning Strategies; Programming; Search Strategies; *Students; Use Studies
IDENTIFIERS	Browsing; Learning Environment; Navigation (Information Systems)

ABSTRACT

This document presents a theoretical approach and a methodology for analyzing data from students interacting with and learning from hypermedia systems. Interactions are mutually influenced by individual students' goals and strategies and the actions supported by the interface of the learning environment. The approach is illustrated by modelling data from an empirical study in which students browsed through a hypertext instructional environment to learn about programming concepts. By using the explanatory power of the computational model, interactions can be analyzed to determine patterns of use. Results obtained from this method of analysis yield specific feedback on system design and prescriptions for improving the design. More theoretically, they provide insights on the nature of human cognition and learning in the context of interactive educational technologies. There are a number of implications for design: (1) it is important to provide high-level navigational aids within hypermedia; (2) designers need to encourage the development of alternate browsing strategies; (3) instructional examples, which are highly valued by students, are important; (4) hypermedia interfaces should also function as memory aids, so as not to interfere with the learning process; and (5) singleton production rules and the uninterpretable cluster should be kept to a minimum. (Contains 15 references.) (AEF)

A Methodology for Analyzing Students' Interactions within Educational Hypertext

Margaret M. Recker
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280 U.S.A.
email: mimi@cc.gatech.edu

Abstract: We present a theoretical approach and a methodology for analyzing data from students interacting with and learning from hypermedia systems. In our approach, interactions are viewed to be mutually influenced by individual students' goals and strategies and the actions supported by the interface of the learning environment. The approach is illustrated by modelling data from an empirical study in which students browsed through a hypertext instructional environment to learn about programming concepts. By using the explanatory power of the computational model, interactions can be analyzed to determine patterns of use. Results obtained from this method of analysis yield specific feedback on system design and prescriptions for improving the design. More theoretically, they provide valuable insights on the nature of human cognition and learning in the context of interactive educational technologies.

Designers of hypermedia systems can never completely anticipate how people will use their systems. This is especially true in educational settings where, typically, users are both domain and hypermedia novices. In these situations, learners lack both a mental model of the domain represented in the system and effective browsing strategies. Therefore, empirical evaluations of system design become paramount. Ideally, such evaluations will provide concrete prescriptions for improving the design. Unfortunately, the identifiability problem is often difficult. Even a cursory review of evaluations of educational technologies reveals a cloudy, complex picture. Empirical results have shown that these learning systems are often used in ways that are completely unintended by its designers, are unproductive, result in aptitude-treatment interactions, and show large individual differences (Jonassen and Mandl, 1990; Steinberg, 1989). Compounding this problem is the fact that it is often difficult to attribute empirical results back to specific features in the design.

In this paper, we present a theoretical approach and a methodology for analyzing student data as they interact with and learn from hypermedia systems. In our approach, interactions are viewed to be mutually influenced by individual students' goals and strategies and the action opportunities supported by the interface of the learning environment (Kirlik, 1993; Pirolli and Wilson, 1992). In our method, which draws upon techniques from both artificial intelligence and the cognitive sciences, student interactions are modelled in terms of both student goals and environmental constraints on action. By using the explanatory power of the computational models, interactions can be analyzed to determine patterns of use. Results obtained from this method of analysis yield specific feedback on system design and prescriptions for improving the design. More theoretically, they provide valuable insights on the nature of human cognition and learning in the context of interactive educational technologies.

The methodology is illustrated by using data from an empirical study in which students browsed through a hypertext instructional environment to learn about recursive programming concepts in Lisp. Following our method, the data are modelled by coupling simulations of individual students' actions with a model of action opportunities supported by the interfaces of the instructional environment. The resulting simulations capture the interactions observed in the empirical data. Specifically, for each student, the corresponding simulation is required to perform students' mouse actions and their verbal utterances, in exactly the same temporal order that these were observed in the data.

The simulations of individual students are analyzed using a statistical clustering algorithms in order to identify common usage patterns across students. These patterns, we argue, represent students' strategies for navigating and learning within the hypertext environment. The explanatory power of our cognitive model provides the semantics for interpreting these patterns in terms of successful and unsuccessful design features of the hypertext system. In addition, it yields specific prescriptions for improving the design.

The remainder of this paper is organized as follows. In the next section, we briefly describe the hypertext learning system and its empirical evaluation. We then present our modelling methodology, and show how the model is used to identify patterns of learner use. We close with a discussion of how an understanding of these patterns can help inform system re-design.

System Description and Empirical Evaluation

Elsewhere we have described in detail the design of a hypertext learning environment, called the Explanation Environment, which contains instructional materials (text and examples) on programming recursion in Lisp (Recker and Pirolli, in press). Briefly, in the Explanation Environment, instructions can be browsed in a non-linear fashion. The environment also contains instructional examples, which are annotated with explanatory elaborations that students can choose to view by clicking with on a button. With this design feature, we hope to provide extra, optional explanations to students who are unable to produce them on their own; students who are able to generate their own explanations for the examples can choose to ignore these additional textual elaborations.

Students are provided with two navigational methods for moving between top-level screens of the instruction. The first, *global navigation*, provides learners with a map of top-level topics, each listed on a button. The second navigational method, *local navigation*, is implemented by providing two buttons on each instructional screen, which learners can click on to move to the next and previous top-level instructional topic, respectively.

An empirical study was conducted in order to examine the effects of the hypertext system on students' initial understanding and their subsequent performance while programming recursion. In this study, students went through five lessons on programming in Lisp. Each lesson had two parts: studying instructional material (learning), followed by programming (problem solving), using an intelligent tutoring system for Lisp, the CMU Lisp Tutor (Anderson et al., 1990). For the target lesson, the lesson on recursion, two sets of computer-based instructions were developed. Students were randomly assigned to one of the two environments to learn about the concepts of recursion prior to programming recursion with the Lisp Tutor. The first environment was the Explanation Environment. A second, control instructional environment was also implemented, which mirrored more standard, linearly structured instruction. Students navigated through both environments by clicking on buttons. These mouse actions provided additional data on students' strategies for learning from instruction.

When we contrasted subjects' performance while programming with the Tutor, we did not find any significant differences in outcome between subjects using the hypertext-based instructional system and those in the control. However, we did find a significant aptitude-treatment interaction. Post-hoc analyses showed that the higher-ability subjects (those that performed well in the pre-intervention lessons) in the hypertext condition made significantly less errors while programming than the low-ability subjects. Subjects in the control condition did not show significant ability-based differences. The full results of the study are presented elsewhere (Recker and Pirolli, in press). In this paper, we focus on the models of student interaction in the hypertext condition and show how we analyzed these to identify common patterns of use.

Modelling Technique

Data from the empirical study form the motivation for a computational model, called SURF (Strategies for Understanding Recursive Functions). SURF is implemented within the Soar architecture (Laird et al., 1987). Soar is an AI production system architecture in which problem solving is carried out by search through problem space in order to achieve particular goals. Soar also includes an experience-based learning mechanism, called *chunking*, which summarizes problem solving experiences into a more efficient form.

In the interest of space, we can only present an overview of the SURF model. More details can be found elsewhere (Recker and Pirolli, in press). The primary goal of the SURF model is to model the learning behavior of individual students in terms of two criteria. The first criterion requires that *every* mouse clicking action by all students is simulated in the exact order that it occurs in the data. This forms what is called the *fine* modelling criterion. The verbal explanations that students made to themselves (they were asked to provide concurrent verbal protocols), which we call self-explanations (Chi et al., 1989; Pirolli and Recker, in press), form the secondary, *coarse* modelling criterion. This meant that students' self-explanations are modelled at a rough level, in the sense that their exact natural language statements are not simulated. More specifically, at each screen in the instructional environments, a student can attempt to self-explain the instruction. The students' verbal protocols are consulted to determine when such self-explanations are exhibited. At each of these instances, the corresponding simulation applies what is called the *comprehension* operator. The application of this operator results in the creation of chunks, representing newly acquired knowledge. In sum, the Surf model, focuses on exactly capturing the temporal sequence of students' interface interactions and their self-explanations.

In the SURF model, student-environment interactions are modelled as two components: (1) a simulation of the possible actions support by the instructional interface, and (2) simulations of individual students' interaction and learning strategies, and their prior knowledge (jointly called *capabilities*).

The interface of the instructional environment is a Lisp simulation of the buttons and instructions that are displayed in each screen context. These buttons and instruction snippets represent opportunities for actions. The presence of buttons offer mouse-clicking opportunities, while instructional snippets present self-explanation opportunities.

Learners' Capabilities. The second component of the model simulates individual learners' capabilities. In Soar, a set of production rules is created for each student, called the learner's *profile*, which represents each student's learning strategies and prior knowledge. Each student's profile is implemented such that when it is loaded in with the interface model, the resulting Soar run fits that student's behavior. Recall that the fit has to meet two criteria: the *fine* criterion requires that every mouse clicking action is captured occurs and the *coarse* criterion models students' self-explanations at a rough grain-size.

Two kinds of methods are currently implemented for modelling learners' capabilities. First, a set of production rules represents the learner's prior knowledge that is used to generate a self-explanation. A second set of production rules represents how the learner selects among possible available actions (or operators). In Soar, the desirability or acceptability of possible alternatives is described in terms of a fixed language of *preferences* (e.g., best, better, reject). In SURF, preference productions are used to express the value of available operators (e.g., selecting a particular button is desirable) in order to simulate a student's actions in the order that these occur. Since preference productions deliberately choose among available operators (and thus are knowledge about knowledge), they can be seen as representing strategic or metacognitive knowledge.

Model Analysis

The preference production rules in a student's simulations models that student's mouse clicking actions and (roughly) verbal explanations, in the exact temporal order that these occurred. These preference productions therefore represent a student's strategic knowledge for explicitly choosing among available actions. In short, they are tangible representations of learning and navigating strategies.

The profiles can be analyzed in order to identify patterns of common use across students. In order to accomplish this, we constructed a dichotomous correlation matrix of all preference production rules that occurred more than once in the union of all student profiles. The resulting matrix of 26 production rules is used to perform a multidimensional scaling, using two dimensions.

As can be seen in the resulting plot Figure 1, several clusters of related of production rules, shown as letters, are evident. These clusters can be interpreted by using the semantics of the cognitive model

```
DIMENSION  2
         -+--------------+--------------+--------------+--------------+-
        2 +                                                           +
          |                                                           |
          |                                                           |
          |                                                           |
        1 +                                      I                    +
          |                       R        HJ     D  C                |
          |           N       PU  ,                                   |
          |          S                    AE     G F                  |
        0 +          Q       T    O           M                  K    +
          |           V                                         L     |
          |                      X                                    |
          |                                                           |
       -1 +                     W        Y                            +
          |                                           B               |
          |                                                           |
          |                                                           |
       -2 +                                                           +
         -+--------------+--------------+--------------+--------------+-
         -2             -1              0              1              2
                                   DIMENSION  1
```

Figure 1: Multi-dimension scaling of preference production rules in students' profiles

(expressed in terms of interface primitives), as follows:

- Group A, E. This cluster contains production rules for navigation using the top-level "Map of Topics" global navigation facility.
- Group H, J. This cluster contains production rules that express a preference for viewing the instructional examples, when available.
- Group D, C, I. This cluster contains production rules that express a preference for a forward, serial navigation strategy and for avoiding backtracking.
- Group F, G. Similar to the second cluster, this cluster contains production rules that express a preference for *studying* instructional examples, especially in lieu of textual instructions.
- Group U, P, T. This cluster contains production rules that involve screen management, specifically for keeping track of screens that are skipped. Since these productions keep track of screens, they make high working memory demands and, as such, probably impose an added cognitive demand.
- Group N, S, Q, V. This cluster is similar to the previous cluster (and occupied the same general area). These production rules keep track of screens that have been viewed and guard against viewing them again. As in the previous cluster, these production rules make high working memory demands.
- Group K, L. These two production rules appear unrelated and it remains unclear why they are clustered.

Implications for Design

Based on this analysis, several conclusions can be drawn about specific features of the hypertext system. Some of these conclusions pertain to the design of educational technology and some apply to understanding how students learn in the context of educational technologies.

First, the existence of a distinct global navigation cluster suggests that the tool is an important and relied-upon facility in the hypertext system. Furthermore, the ability to use this facility appears to be a separate, identifiable cognitive skill. Taken together, these imply the importance of providing high-level navigational aids within hypermedia.

Second, as a default browsing strategy, novices appear to prefer forward, serial browsing and to dislike backtracking. Such preferences have been found in other evaluative studies of hypermedia systems (Mayes et al., 1990), and may reflect a default, novice navigation strategy. Designers should ensure this default strategy is close to optimal when designing educational hypermedia. Additionally, designers ought to consider scaffolding methods to support and encourage the development of alternate browsing strategies.

Third, instructional examples appear to be highly valued by students. Students both preferred to select instructional screens containing examples and to study these examples. In fact, the reliance by novices on examples in the early phases of learning a new domain is a robust finding in the literature (LeFevre and Dixon, 1986; Pirolli and Anderson, 1985; Ross, 1984; Sweller and Cooper, 1985).

Fourth, the clustering of production rules that keep track of visited screens suggests that the hypertext environment does not possess adequate features for helping students mark their current location within the system, mark where they've been, mark what they've chosen to skip and determine what remains to be seen. As a result, learners have to do much of this bookkeeping for themselves, which possibly adds undue cognitive load and may interfere with learning. This highlights the importance of designing hypermedia interfaces that they also function as external memory aids.

Finally, what are we to make of singleton production rules (those that did not cluster) and the uninterpretable cluster? The singletons represent isolated behaviors, reflecting the myriad of ways that students can choose to interact with systems. The uninterpretable clusters represent patterns that are not accounted for by the interface. In short, they are unexpected behaviors. In general, it is unlikely that system design will ever completely eliminate these. We can only hope to minimize their occurrences, relative to the explainable patterns.

Discussion

In this paper, we have presented a framework and methodology for analyzing students' interactions and learning in the context of using a educational hypertext system. The framework posits interaction to be mutually influenced by environmental constraints (i.e., actions supported by the interface) and individual learner goals and strategies. The methodology, which relies on an analysis of a model of observed interaction, yields specific results for evaluating system design. These results also serve as prescriptions for improving the design.

Clearly, before claiming generality, the overall approach should be applied to other data of students interacting with educational technology. In fact, we are currently planning to use our method with data collected as students learned to troubleshoot a complex system through using a simulation coupled with an intelligent tutoring system, called Turbinia-Vyasa (Vasandani and Govindaraj, 1993).

In closing, it could be argued that this is an expensive methodology, since it requires the formulation and implementation of computational cognitive models. However, we note that the use of Soar is not a prerequisite. Any cognitive architecture in which units of skill and interface primitives can be represented will suffice. In addition, the time to implement the computational models should decrease with growing expertise. However, in closing, we argue that it will never be possible to completely obviate the need for a deep of understanding of student characteristics and interface capabilities in order to appreciate the myriad of ways students can adapt to educational technology.

References

Anderson, J., Boyle, C., Corbett, A., and Lewis, M. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42(1):7–49.

Chi, M., Bassok, M., Lewis, M., Reimann, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182.

Jonassen, D. and Mandl, H., editors (1990). *Designing Hypermedia for Learning.* Springer Verlag, Berlin.

Kirlik, A. (1993). Modeling strategic behavior in human-automation interaction: Why and "aid" can (and should) go unused. *Human Factors*, 35(2):221–242.

Laird, J., Rosenbloom, P., and Newell, A. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.

LeFevre, J. and Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction*, 3:1–30.

Mayes, T., Kibby, M., and Anderson, T. (1990). Learning about learning from hypertext. In Jonassen, D. and Mandl, H., editors, *Designing Hypermedia for Learning*, pages 227–250. Springer Verlag, Berlin.

Pirolli, P. and Anderson, J. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39(2):240–272.

Pirolli, P. and Recker, M. (in press). Learning strategies and transfer in the domain of programming. *Cognition and Instruction.*

Pirolli, P. and Wilson, M. (1992). Measuring learning strategies and understanding: A research framework. In *Proceedings of the International Conference on Intelligent Tutoring systems*, Berlin. Springer Verlag Lecture Notes in Computer Science.

Recker, M. and Pirolli, P. (in press). Modelling individual differences in students' learning strategies. *Journal of the Learning Sciences.*

Ross, B. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology*, 16:371–416.

Steinberg, E. (1989). Cognition and learner control: A literature review, 1977-1988. *Journal of Computer-Based Education*, 16(4):117–121.

Sweller, J. and Cooper, G. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 7:1–39.

Vasandani, V. and Govindaraj, T. (1993). Knowledge structures for a computer-based training aid for troubleshooting a complex system. In Towne, D., de Jong, T., and Spada, H., editors, *The Use of Computer Models for Explication, Analysis, and Experiential Learning.* NATO ASI Series F, Springer Verlag, Heidelberg.

Acknowledgements

This research was supported by NSF under contract IRI-9001233 to Peter Pirolli and by a University of California Regents' Dissertation Fellowship. Preparation of this manuscript was supported by the Army Research Institute under contract No. MDA-903-90-K-0112 to Janet Kolodner.